AE 353 Project 1: Control Moment Gyroscope

Anshuk Chigullapalli[1]
University of Illinois at Urbana-Champaign, Champaign, Illinois, 61820, USA

I. Abstract

The goal of this project is to create a feedback controller that actuates a simulated single-gimbal control moment gyroscope, to control the direction of a spacecraft "platform". The report also outlines the additional benefits and drawbacks of the designed controller. The controller and the simulation are designed, implemented, and tested in a Jupyter Notebook python environment.

II. Nomenclature

CMG	=	Control Moment Gyroscope
f	=	The equations of motion as a function of state and inputs
q_1	=	Angle of the platform
q_2	=	Angle of the gimbal
q_3	=	Angle of the rotor
τ_2	=	Torque applied to the gimbal
τ_3	=	Torque applied to the rotor
u	=	The vector of inputs of the system
v_1	=	Angular velocity of the platform
v_2	=	Angular velocity of the gimbal
v_3	=	Angular velocity of the rotor
x	=	State vector

III. Introduction

The goal of this project is to create a robust controller for a control moment gyroscope that controls the angular position of a spacecraft platform. Control moment gyroscopes are used frequently in spacecraft to control their attitude. An example of a control moment gyroscope is shown in Fig. 1. Although they do have a few flaws, such as singularities and saturation, they do have considerable benefits as they provide major propellent savings.[2] [1]

Fig. 1: A CMG used on the International Space Station

[1] Student, Department of Aerospace Engineering.
[2] Wikipedia – Control Moment Gyroscope: https://en.wikipedia.org/wiki/Control_moment_gyroscope

The CMG model in this project isn't a 3-axis CMG like the ones used in spacecraft. Instead, it is a Single-Gimbal CMG, that can only change the angle of the platform in one plane. A picture of the system used in this project is shown in Fig. 2.

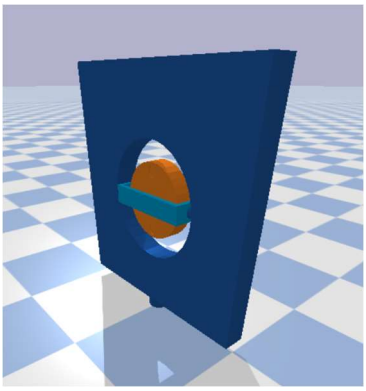

Fig. 2: CMG and Spacecraft Platform

The sections of this report go over the design process involved in developing a feedback controller for this system. As an arbitrary test case, the controller is designed to turn the platform to a desired angle of 45 degrees.

$$q_{1_{des}} = 45\,^{\circ}$$

It then discusses the implementation of this control system in the provided simulation tool. Though the controller was designed to meet the requirements of the project, there are a few additional features of the controller that are particularly interesting. On the other hand, the controller also has flaws that can be further explored and corrected.

IV. Model

A. Equations of Motion

To be able to design a controller, we first need to develop a state space model of the system. However, this system is highly non-linear, as seen in the set of equations shown below[3].

$$\dot{v_1} = -\left(\frac{5 \cdot (200\tau_3 \cdot \sin(q_2) + \sin(q_2)\, v_1 v_2 + 2\cos(q_2)\, v_2 v_3)}{10(\sin(q_1))^2 - 511}\right)$$

$$\dot{v_2} = \frac{10}{11}(100\tau_2 - \cos(q_2)\, v_1 v_3)$$

$$\dot{v_3} = -\left(\frac{51100\tau_3 + 5\sin(2q_2)\, v_2 v_3 + 511\cos(q_2)\, v_1 v_2}{10(\sin(q_2))^2 - 511}\right)$$

These equations of motion require linearization about an equilibrium point. That process is described in Section B.

Before linearization, the state and the input matrices must be chosen. This brings us to the first design choice of the project, picking the variables for the state space model. I experimented with different versions of the state matrix, each with a different set of variables. However, I realized that only the version of the state space model that included the effects of the torque on the rotor guaranteed stability with minimal tuning, even when trying to control the speed of the rotor. This motivated the choice of the state and input matrices shown below.

[3] These equations are from the project description on the course website: https://tbretl.github.io/ae353-sp21/projects#report

$$x = \begin{bmatrix} q_1 \\ v_1 \\ q_2 \\ v_2 \\ v_3 \end{bmatrix}$$

The derivative of this matrix, i.e. $\dot{x}$, is shown below.

$$\dot{x} = \begin{bmatrix} v_1 \\ \dot{v_1} \\ v_2 \\ \dot{v_2} \\ \dot{v_3} \end{bmatrix}$$

The position of the rotor (q_3) is not of major concern in the design of the controller, since the rotor is symmetrical about the axis in which it is spinning and its only purpose is in v_3 This is why it is not included in the x vector. v_3 was included because, as will be seen in the following sections, the rotor speed is controlled by the feedback controller to alter the rate at which the controller reaches the desired angle.

There are only two inputs in the system, and they are placed in the u vector. τ_2 is the torque applied on the gimbal and τ_3 is the torque applied on the rotor.

$$u = \begin{bmatrix} \tau_2 \\ \tau_3 \end{bmatrix}$$

$$\dot{x} = Ax + Bu$$

This means that A is a 5×5 matrix and B is a 5×2 matrix. Since there is no Kalman filtering or signal processing of any kind involved, a choice for y is not required. Hence, the second equation of the state space model (C and D) is not set.

To find the values of A and B, we have to linearize the system about an equilibrium point.

B. Linearizing the Model

To linearize the system, we put together all the equations of motion as a function $\dot{x} = f(q_1, v_1, q_2, v_2, v_3, \tau_2, \tau_3)$

$$\dot{x} = f = \begin{bmatrix} v_1 \\ -\left(\dfrac{5 \cdot (200\tau_3 \cdot \sin(q_2) + \sin(q_2)\, v_1 v_2 + 2\cos(q_2)\, v_2 v_3)}{10(\sin(\mathrm{q}_1))^2 - 511}\right) \\ v_2 \\ \dfrac{10}{11}(100\tau_2 - \cos(q_2)\, v_1 v_3) \\ -\left(\dfrac{51100\tau_3 + 5\sin(2q_2)\, v_2 v_3 + 511\cos(q_2)\, v_1 v_2}{10(\sin(q_2))^2 - 511}\right) \end{bmatrix}$$

1. Equilibrium Point

The system needs to be linearized about an equilibrium point. An equilibrium point is a set of values for the inputs of f that make $f = 0$. The points below describe the choice of equilibrium values:

1) From the function, it is clear that $v_{1_e} = v_{2_e} = 0$ must be part of the equilibrium values.
2) $\tau_{2_e} = \tau_{3_e} = 0$ is also a requirement for the equilibrium point.
3) q_1 is not in any of the equations so its equilibrium point value can be anything. So, $q_{1_e} = 45$ degrees was chosen to match the desired angle.
4) Although it can also be anything, $q_{2_e} = 0$ to keep the equilibrium point reasonable and equal to the starting and final positions of the gimbal.
5) v_3 can also be anything, but since we're starting off at 100 rpm, I set it as that.

2. Calculating A and B

The Jacobian of f is calculated with respect to the variables of the state and input matrices to create functions for A and B.

$$\boldsymbol{J}_f(q_1, v_1, q_2, v_2, q_3) = A(q_1, v_1, q_2, v_2, q_3)$$
$$\boldsymbol{J}_f(\tau_2, \tau_3) = B(\tau_2, \tau_3)$$

These are then evaluated at the equilibrium point. The result is the values of of A and B at the set equilibrium point. This process was implemented in code. The resulting A and B are shown here.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.20493 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -9.52 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 90.9090 & 0 \\ 0 & 100 \end{bmatrix}$$

With this, we have linearized our system. Now, we can begin designing the controller for this system and using the linearized dynamics to calculate stable values for K.

C. Calculating Controller Gains

We still have not covered a method to efficiently calculate working K gains. Therefore, here we resort to looping through random numbers until a set of controller gains resulted in a stable response. This section will go over the method used to check if the controller creates a stable response, and if it does, whether than response is fast enough. In a feedback loop, the input u is a function of the state x.

$$u = -Kx$$

This implies that K is a 2×5 matrix. Substituting this equation back into the state space equation, we get:

$$\dot{x} = (A - BK)x$$
$$\dot{x} = Fx$$

In the above equation, $F = A - BK$. Note that F is **not** the same as f, which was used previously in the process. We now have a basic differential equation whose solution is an exponential function.

$$x(t) = e^{Ft}x(0)$$

Here, e^{Ft} is the matrix exponential of Ft and not a scalar exponential. Our goal is for $x(t)$ to converge to 0, and for this to happen, the eigenvalues of F must have a negative real part. Therefore, the verdict on whether K gives us a stable or unstable response depends on the sign of the real part of the eigenvalues of F. Another criteria that I personally set as a design choice was to constrain the imaginary part of the eigenvalues of F to be zero. This way, there is no oscillation in $x(t)$.

To quickly test many values of K to get valid results, a loop was set in python that kept testing K with random values (found using numpy.random.rand() function) until a set of values is found that results in F in having eigenvalues with negative real parts and zero imaginary parts.

$$K = \begin{bmatrix} random & random & random & random & 0 \\ 0 & 0 & 0 & 0 & random \end{bmatrix}$$

This was done for two reasons. The first was to simplify the process of finding a useful set of K gains. Having τ_3 as a function of all the state variables seemed unnecessary and so it was set to be only a function of the rotor speed.

Similarly, τ_2 was kept independent of the rotor speed. This design choice essentially allowed for stable control of the rotor speed, while also keeping the controller insulated from the potential negative effects of having to control too many interdependent states.

However, eigenvalues with a negative real part only indicate that the system is stable. It does not, however, guarantee that the system will approach the desired values within a small time frame, and so doesn't guarantee that the controller is useful. Therefore, we need to calculate the response $x(t)$ and plot it to see whether the response is useful. Jupyter notebook was having trouble directly computing the matrix exponential e^{Ft}. So F had to be diagonalized. Therefore, we use the following relation to calculate the response instead.

$$x(t) = Ve^{St}V^{-1}x(0)$$

Where V contains the eigenvectors of F, and S is a diagonal matrix with F's eigenvalues as its elements. Since S is a diagonal matrix, computing the matrix exponential is a lot easier. With this, we can test the controller even before actually running it in the simulation.

Shown below is an example of when the controller gains were stable (all eigenvalues have negative real part), but the response was not quick enough.

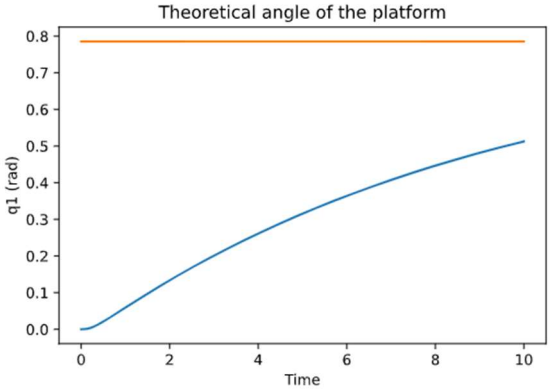

Fig. 3: Stable but slow controller

In comparison, here is the response from the controller that is finally used.

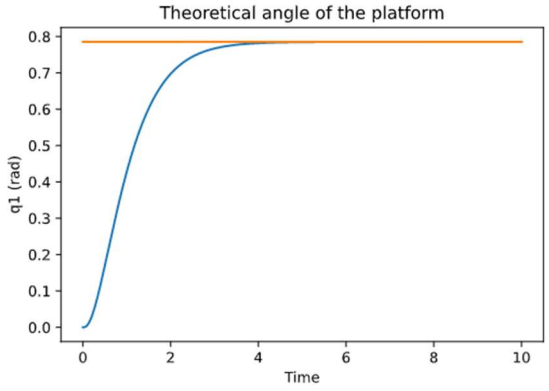

Fig. 4: Stable and Fast Controller

Clearly, the controller response shown in Controller Fig is very good and can be implemented in the simulation. We shall move forward with this set of values for K, shown below. With this set of gains, all the eigenvalues of F have a negative real part and zero imaginary part.

$$K = \begin{bmatrix} 0.8086 & 0.1356 & 0.7489 & 0.2595 & 0 \\ 0 & 0 & 0 & 0 & 0.9166 \end{bmatrix}$$

V. Results

The controller gains derived in the previous section were implemented in a feedback controller in the given simulation. The parameters in the simulation are set as follows.

1) $q_{1_{des}} = 45°$. This is the desired angle of the platform.
2) $v_{3_{des}} = 50 \frac{rad}{s}$. This is to make the rotor faster, and so help the system reach its desired state faster.
3) The desired values of the states are set to 0.
4) The initial condition is $v_3 = 100\ rpm = 10.472 \frac{rad}{s}$. All other initial conditions are zero.
5) There is no damping.
6) The time step of the simulation is 0.001 s (1 ms)

The results of the simulation are shown in Fig 5. A few major things to note about these results:

1) The platform angle quickly converges to the desired angle of 45 degrees. This is the result that we wanted and it is very good.
2) The outer angular velocity converges to zero near the desired angle appropriately. The gimbal angle follows a similar trend.
3) The rotor speed quickly increases to the commanded 50 rad/s. This helps the system reach its final state faster. However, it does result in the rotor torque exceeding the maximum joint torque. Ideally, this would be optimized so that the command doesn't exceed the torque. This is discussed further in later sections.
4) We can also compare the result of the simulation with the expected result (from Fig. 4) calculated with the linearized system. There is a slight difference because the linearized system will never be the same as the true non-linear system, but the results are surprisingly similar.

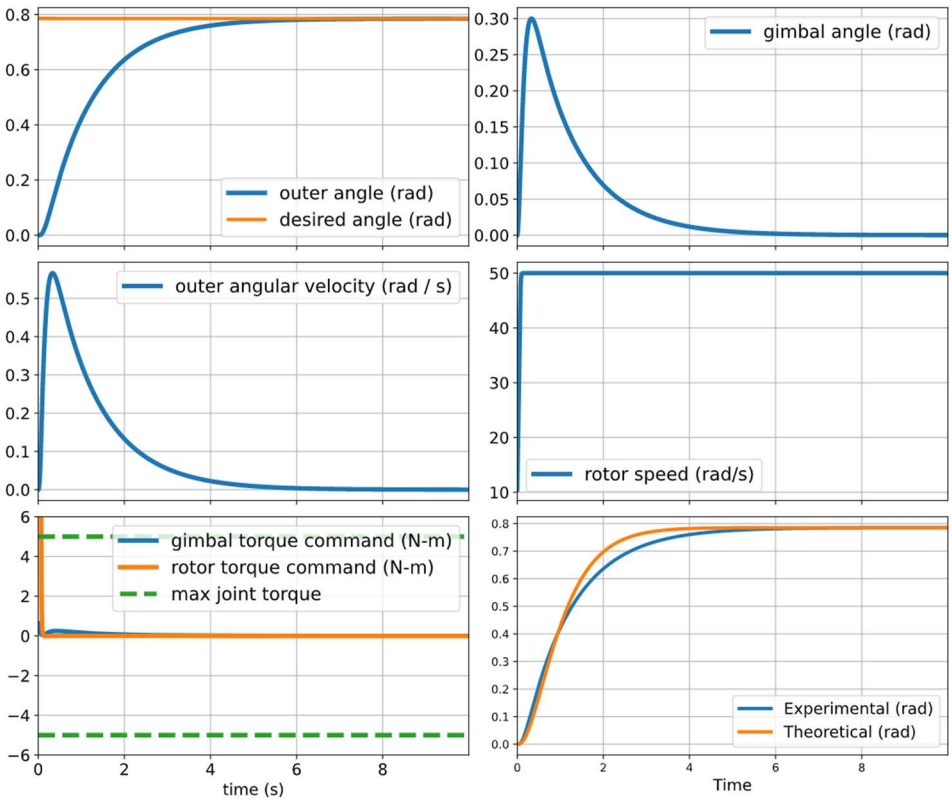

Fig 5: Results of Simulation

Overall, this is a great result, and the system has achieved its goal of rotating the platform to the desired angle. Next, we will explore how robust this controller is, and whether it can handle different test cases.

VI. Further Testing

A. Different Desired Platform Angles

The controller is also advanced enough to quickly converge to any desired q_1 between $-\pi$ and π radians. The plots below show the result of the simulation when the desired angle $q_{1_{des}} = 180°$.

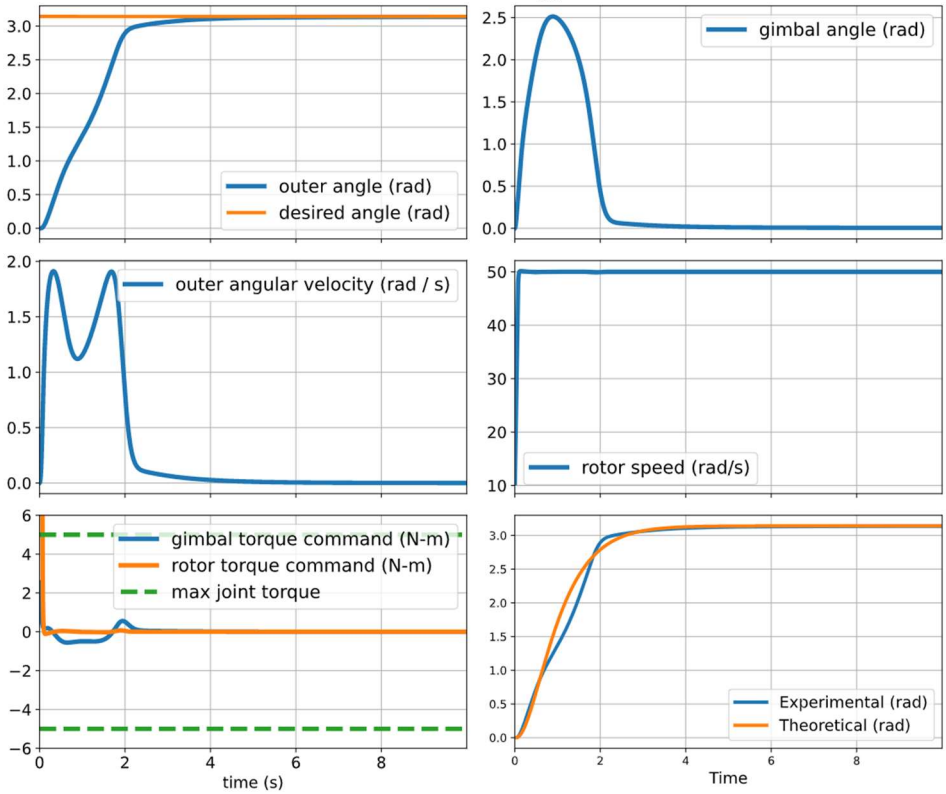

Figure 6: Simulation with desired angle at 180 degrees

We begin to see interested behavior! The system still converges to its desired value in a very short period of time, which is great. But we notice that the angular velocity of the platform isn't a clean curve as before. The reason for this is apparent when we compare the experimental and theoretical results. In the linear system, the angle of the gimbal can be increased for a greater turning action on the platform. It is linearly proportional. This is not the case in the real simulation! When the angle of the gimbal becomes greater than 1.57 radians (90 degrees), the turning action starts decreasing. This is also the reason why this system stops working when the desired angle is greater than ~210 degrees. This is why if the system needs to be actuated, the desired angle should be kept in the range of $-\pi$ to π. The gimbal turns too much, and the platforms starts moving in the opposite direction. This is a major non-linear characteristic of the equations of motion that we lose when we linearize the system.

B. Gimbal Lock

I decided to explore the non-linearity of the system further and realized that there is a particular desired angle for which the gimbal angle becomes exactly π radians and the gimbal velocity becomes 0, so the system just stops moving. This phenomenon, I found out, is called gimbal lock[4]. After testing many values, I noticed that this occurs when

[4] What is gimbal lock as it pertains to spacecraft, and why is it such a big deal? Link: https://www.reddit.com/r/askscience/comments/27tacb/what_is_gimbal_lock_as_it_pertains_to_spacecraft/

$q_{1_{des}} \approx 209.6\,°$. Even more fascinating is this: when the gimbal locks, the platform stops at exactly 45°, which was the equilibrium point about which the system was linearized! It seems like the gimbal lock position occurs when the system reaches an unstable equilibrium. Amazing!

VII. Conclusion

The feedback controller achieved its goal of turning to the desired angle of 45 degrees in a small timeframe. The calculated K gains gave stable results and the theoretical and actual results are in good agreement. Further testing also showed that the system responded well to different desired angles, although angles of magnitude greater than 209 degrees broke the controller. Although not mentioned in the report, there were a few tests done with different initial conditions, but the controller was unable to handle them. That leaves scope for future experimentation.

Acknowledgements

I'd like to gratefully acknowledge the instruction of Professor Timothy Bretl and teaching assistant Jacob Kraft, who provided the guidance and material to successfully complete this project. Their tutorials on the homework and in the class formed the basis for the procedure used in the design of this controller. Their future guidance and feedback will be used to improve this report. I'd also like to acknowledge the help of my peers on the online discussion forum. I'd like to specifically mention Alan Hong for his code of the random number loop.

References

There are no additional references for this report. Website references are listed in the footnotes.